5可靠传输和首部

2019年6月18日 13:39

-
- 可靠传输的工作原理

1. 理想的传输条件：传输信道不产生差错；接收方总是来得及处理数据

1- 停止等待协议

(1) 早期数据链路层也会使用停止等待协议来保证尽量可靠

(2) TCP的等待方法复杂得多，以后的笔记以单向发送的情况做说明

1. 无差错情况

(1) 发送完一个分组（指TCP报文段）后暂停等接收方的确认消息

(2) 收到确认后才发送下一个分组

2. 出现差错

(1) 差错指接收方B检测到数据差错或未接收到丢失的分组

(2) 发送方A为发送的分组设置一超时计时器，若规定时间内未收到B的确认消息，则默认分组传送出现差错，重发该分组，直至收到确认

(3) 为防止网络延迟使B收到误以为丢失的重复发送的分组，A需给分组编号

(4) 为了能重传，发送后仍需保存副本，待收到确认后再清除

(5) 具体等待时间要考虑拥塞，时延，必须大于RTT，但又不适合太大

3. 确认丢失和确认迟到

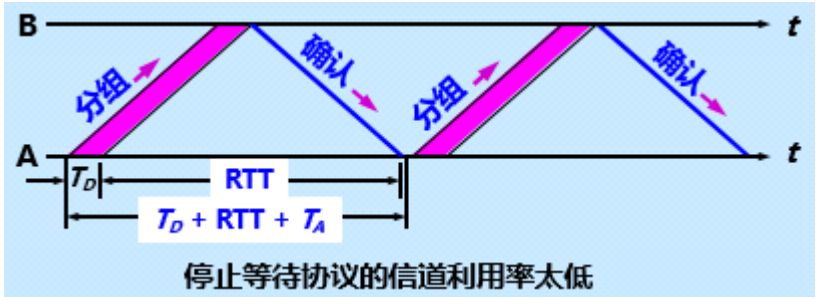
(1) B连续收到两个序号为M1的分组，首先要丢弃该分组，即不向上层交付

(2) 之后要重发确认，因为上一次确认可能丢失了，需要重新确认

(3) 若A连续收到两个对M1的确认，则说明是前一个说明迟到了，需忽略后一个；若反复收不到确认，说明通信线路实在太差

(4) 因而该协议又称为Automatic Repeat reQuest自动重传请求ARQ，因为任何情况下接收方都不会主动请求重传某一分组，全靠发送方自行猜测

4. 信道利用率

(1) 

(2) 信道利用率U=TD/TD+RTT+TA

(3) 其中TD=发送分组时间=数据长度/发送速率；TA=发送确认时间；RTT=分组传送时间+确认传送时间（严格来讲分子应该不包含首部的时间，但一般利用率不考虑那么严格）

(4) 当RTT远大于TD及反复重传时，信道利用率就极低

(5) 为了提高利用率，显然应采用流水线传输，使用连续ARQ和滑动窗口

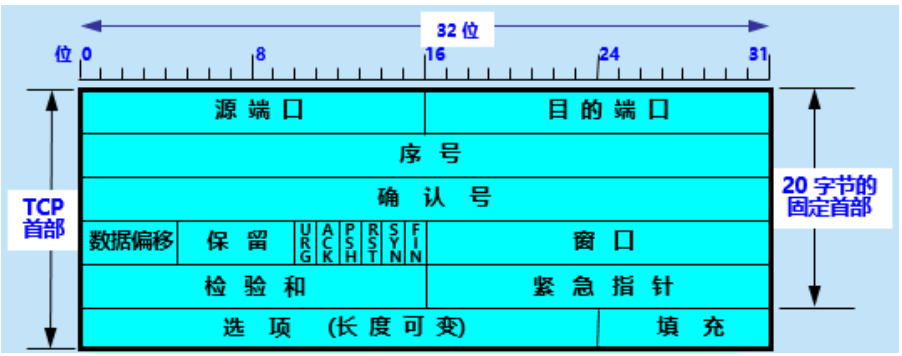
2- 连续ARQ协议

1. 滑动窗口思想：将可发送的首个和末个分组视作窗口首尾
   （1）每收到一个确认，A就把发送窗口向前滑动一下（序号大的方向）
2. B的确认方式一般是累积确认：按序到达的最后一个分组号
3. A的重传方式一般是go-back回退：从第一个B未收到的分组号开始重传
4.

	连续ARQ协议	停止等待协议
发送的分组数量	一次发送多个分组	一次发送一个分组
传输控制	滑动窗口协议	停等-等待
确认	单独确认 + 累积确认	单独确认
超时定时器	每个发送的分组	每个发送的分组
编号	每个发送的分组	每个发送的分组
重传	回退N，多个分组	一个分组

-
- TCP报文段的首部格式

1- 概述

1. 

2. 前20字节是固定的（最小长度），后4n字节是可选项

2- 固定部分

1. 源、目的端口；各2字节
2. 序号seq；4字节；~2^32 - 1；共2^32=4G=4294967296个
3. 确认号ack；占4字节；期待对方下一个报文段的序号；若确认号=N时，表示0~N-1字节的数据都正确收到了
4. 数据偏移；占4位；5~15；数据段的起始位置，即首部长度；单位是32位，说明TCP首部最大长度40字节
5. 保留；占6位；暂时没想好干啥，都置0
6. 紧急URGent；1位；取1时表示有紧急数据，如中断命令，需要插队
7. 确认**ACK**nowledgment；1位；取1时表示是确认报文，确认号才有效
8. 推送PuSH；1位；取1时表示希望对方不等缓存填满即直接向上层交付
9. 复位ReSeT；1位；取1时表示出现严重差错，希望对方释放TCP连接
10. 同步**SYN**chronization；1位；取1而ACK取0时表示这是这是连接请求，取1且ACK也取1时表示是接受连接报文
11. 终止**FIN**ish；1位；取1时表示数据发送完毕，请求释放连接
12. 窗口rwnd；占2字节；0~2^16 -1；表示该报文的发送方的接受窗口大小
13. 检验和；占2字节；类似UDP，加上12字节的伪首部（协议字段不同于UDP的

17，应该是6，如果是IPv6，伪首部也会对应改）

14. 紧急指针；占2字节；紧急位取1时，用于指出紧急数据的字节数，即末尾在报文段中的位置；神秘的是窗口为0时照样可以发紧急数据
15. 选项；长度0~40字节；没有选项的首部长度，即前14项的长度=20字节
16. 填充字段；当首都长度不为4字节整数倍时填充的0

3- Maximum Segment Size最大报文段长度：数据字段的最大长度

1. MSS+首部长度=真实TCP报文段最大长度
2. MSS默认值为536字节（没算上TCP和IP各至少20字节首部）

4- 选项

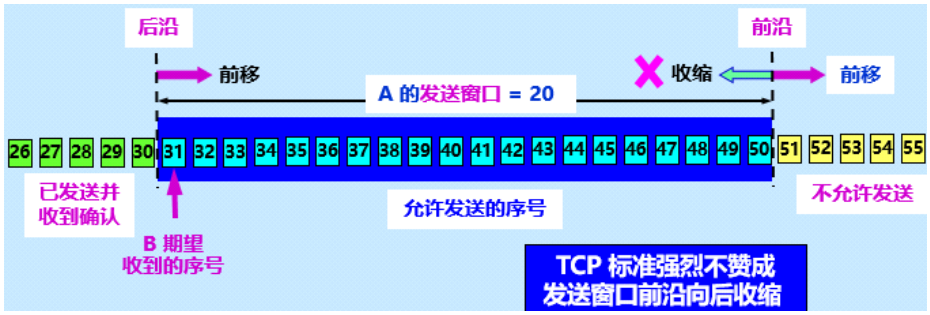
1. 窗口扩大；占3字节；有1字节表示移位值S，表示希望对方窗口大小的位数变成16+S，即0~2^(16+S)-1，S最大值为14
2. 时间戳；10字节；取其中有4字节时间戳值字段和4字节时间戳回送回答字段，用于计算RTT，和Protect Against Wrapped Sequence numbers，即防止序号绕回PAWS：避免序号seq超过2^32时无法判断序号有没有绕回
3. 选择确认SACK，详见下节最后

-
- TCP可靠传输的实现

1- 以字节为单位的滑动窗口

1. 发送窗口：未收到确认的情况下，可以发送的连续字节的数据的范围

(1)

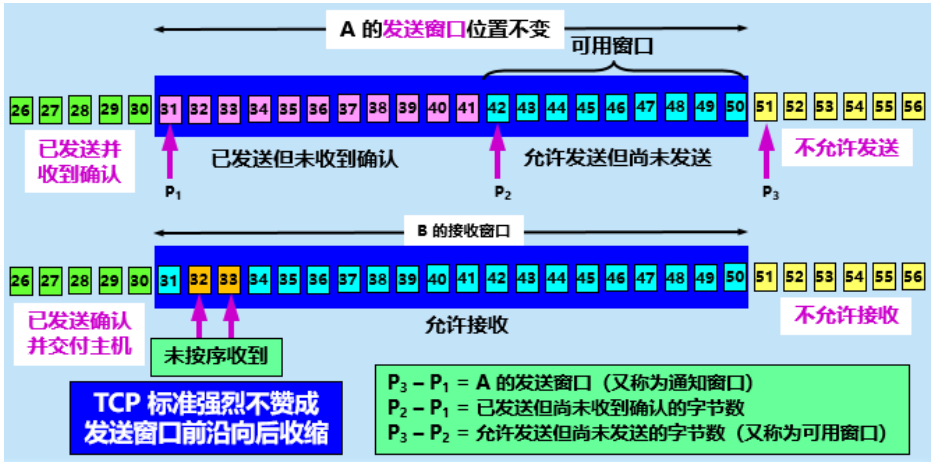


(2) 通常是发送缓存的子集

(3) 大小随对方的接收窗口大小和拥塞情况适当调整

2. 接收窗口：允许接收的数据的范围

(1)



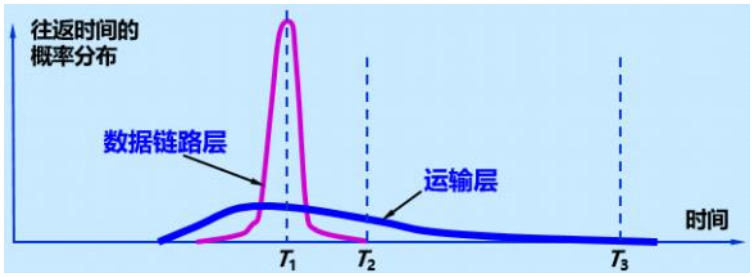
(2) 不按序到达的数据一般也不能交付给上层应用

(3) 连续到达的确认号可以稍晚发送，也可以在给对方传送数据的报文里捎带上，但最好不要晚超过0.5秒，避免不必要的重传

3. 发送缓存
    (1) 暂存打算发的数据
    (2) 暂存发完尚未收到确认的数据
    (3) 发送缓存的后沿和发送窗口的后沿是重合的
4. 接收缓存
    (1) 暂存尚未被程序读取的数据
    (2) 暂存未按序到达的数据

2- 超时重传时间的选择

1. 往返时间RTT

(1) 

2. 平滑的加权平均往返时间RTTs（s指smoothed）
    (1) 新RTTs=（1-α）|旧RTTs + α 新RTT
    (2) α越接近1会使RTT更新越快，RFC6298推荐α=1/8
3. RTTD=RTT偏差的加权平均值，与RTTs和新RTT样本的差的绝对值有关
    (1) 新RTTD=（1-B）|旧RTTD + β | RTTs-新RTT |
    (2) β的推荐值=1/4
4. 自适应算法
    (1) RetransmissionTime-Out超时重传时间RTO，在RFC6298中被推荐为
    (2) RTO=RTTs + 4 x RTTD
5. Karn算法
    (1) Q：重传报文的确认应该按首次发送还是重传发送计算RTT
    (2) A：重传了就不作为RTT样本
    (3) Q：反复重传导致无法更新RTO，再导致更多反复重传怎么办
    (4) A：每重传一次就让RTO乘以γ一次，γ的典型值是2

3- Selective ACK选择确认SACK

1. Q：两段较长连续数据间缺了一个序号的数据，能不能不让后段数据被重传
2. A：可以在首部选项里添加SACK，说明[L,R)才是需要重传的内容
3. 需要在建立连接前讲清楚要不要使用这个功能
4. RFC2018对[L,R)边界格式有详细的规定，但并没有要求对方该怎么处理SACK
    1)
    2)
    3)
    4)
    5)
    6) ----------我是底线----------